Amendments to the claims,
Listing of all claims pursuant to 37 CFR 1.121(c)

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently amended) In a database system, a method for parallel optimization of a query, the method comprising:

generating a plurality of parallel plans for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;

adjusting parallel operators of each parallel plan ~~if necessary~~ based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable;

creating a schedule for each parallel plan indicating a sequence for execution of operators of each parallel plan, wherein the schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query and includes separating a resource intensive operator into a plurality of operators;

determining execution cost of each parallel plan based on its schedule; and

returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query.

2. (Original) The method of claim 1, wherein the query comprises a Structured Query Language (SQL) expression.

3. (Original) The method of claim 1, wherein said generating step includes generating an operator tree for each parallel plan based on the query.

4. (Original) The method of claim 3, wherein said step of generating an operator tree includes generating nodes of the operator tree as iterators for applying predefined behavior to data.

5. (Original) The method of claim 3, wherein said step of generating an operator

tree includes inserting a parallel operator in the operator tree.

6. (Original) The method of claim 5, wherein said step of generating an operator tree includes dividing a query operation into sub-tasks and said parallel operator provides for executing said sub-tasks in parallel.

7. (Previously presented) The method of claim 6, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of storage units.

8. (Previously presented) The method of claim 6, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of CPUs.

9. (Previously presented) The method of claim 5, wherein said parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

10. (Previously presented) The method of claim 1, wherein said generating step includes generating a parallel plan using a partitioning property so as to partition data among operators of the parallel plan.

11. (Original) The method of claim 1, wherein said generating step includes generating a cost vector for each parallel plan.

12. (Original) The method of claim 11, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator in the parallel plan, and resource usage of an operator in the parallel plan for a certain time period.

13. (Previously presented) The method of claim 11, wherein said generating step further comprises:

pruning a first parallel plan having a cost vector costing more in each vector

dimension than a second parallel plan.

14. (Original) The method of claim 1, wherein said generating step includes generating a plurality of parallel plans based at least in part on partitioning and multi-dimensional costing.

15. (Previously presented) The method of claim 1, wherein said adjusting step includes adjusting a parallel plan based on maximum number of threads available at compile time.

16. (Canceled)

17. (Previously presented) The method of claim 1, wherein said step of adjusting parallel operators of each parallel plan further comprises:

adjusting parallel operators based on available memory resources.

18. (Canceled)

19. (Original) The method of claim 1, wherein said creating step includes identifying pipelines in each parallel plan.

20. (Original) The method of claim 19, wherein said creating step includes constructing a pipeline dependency tree based on dependencies among operators of each parallel plan.

21. (Original) The method of claim 20, wherein said creating step includes determining order of execution of pipelines based on the pipeline dependency tree and available resources.

22. (Original) The method of claim 19, further comprising:

if resource usage of a particular pipeline is greater than resources available for the

particular pipeline, splitting the particular pipeline into a plurality of pipelines.

23. (Original) The method of claim 22, wherein said step of splitting the particular pipeline includes adding operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

24. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 1.

25. (Original) A downloadable set of processor-executable instructions for performing the method of claim 1.

26. (Currently amended) A system for parallel optimization of a database query, the system comprising:

a search engine for generating a plurality of parallel plans which can be used for obtaining data requested by the query, the parallel plans including parallel operators for executing portions of the query in parallel;

a parallel scheduler for adjusting parallel operators of each parallel plan ~~if necessary based~~ on maximum number of threads available for executing the query and creating a schedule for the parallel plan indicating a sequence for execution of operators of the parallel plan, wherein the maximum number of threads is user configurable and wherein the schedule is created based upon dependencies among operators of each parallel plan and resources available for executing the query and the parallel scheduler separates a resource intensive operator into a plurality of operators; and

a module for determining execution cost of each parallel plan based on its schedule, and returning a result of a particular parallel plan having lowest execution cost for obtaining data requested by the query.

27. (Original) The system of claim 26, wherein the query comprises a Structured Query Language (SQL) expression.

28. (Original) The system of claim 26, wherein the search engine generates an operator tree for each parallel plan based on the query.

29. (Original) The system of claim 28, wherein the search engine generates nodes of the operator tree as iterators for applying predefined behavior to data.

30. (Original) The system of claim 28, wherein the search engine inserts a parallel operator in the operator tree.

31. (Original) The system of claim 30, wherein the search engine divides a query operation into sub-tasks and said parallel operator provides for executing said sub-tasks in parallel.

32. (Original) The system of claim 31, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of storage units.

33. (Original) The system of claim 31, wherein said parallel operator provides for executing said sub-tasks in parallel across a plurality of CPUs.

34. (Previously presented) The system of claim 30, wherein said parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

35. (Previously presented) The system of claim 26, wherein the search engine generates a parallel plan using a partitioning property so as to partition data among operators of the parallel plan.

36. (Original) The system of claim 26, wherein the search engine generates a cost vector for each parallel plan.

37. (Original) The system of claim 36, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator in the parallel plan, and resource usage of an operator in the parallel plan for a certain time period.

38. (Previously presented) The system of claim 36, wherein the search engine prunes a first parallel plan having a cost vector costing more in each vector dimension than a second parallel plan.

39. (Original) The system of claim 26, wherein the search engine generates a plurality of parallel plans based at least in part on partitioning and multi-dimensional costing.

40. (Previously presented) The system of claim 26, wherein the parallel scheduler adjusts a parallel plan based on maximum number of threads available at compile time.

41. (Canceled)

42. (Canceled).

43. (Original) The system of claim 26, wherein the parallel scheduler identifies pipelines in the parallel plan.

44. (Original) The system of claim 43, wherein the parallel scheduler constructs a pipeline dependency tree based on dependencies among operators of a parallel plan.

45. (Original) The system of claim 44, wherein the parallel scheduler determines order of execution of pipelines based on the pipeline dependency tree and available resources.

46. (Original) The system of claim 43, wherein the parallel scheduler splits a

particular pipeline into a plurality of pipelines.

47. (Original) The system of claim 46, wherein said parallel scheduler adds operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the plurality of pipelines.

48. (Currently amended) A method for parallel optimization of a query requesting data from a database, the method comprising:

creating a plurality of operator trees for executing the query, the operator trees providing for execution of portions of the query in parallel;

adjusting the portions of the query to be executed in parallel based on maximum number of threads available for executing the query, wherein said maximum number of threads is user configurable;

generating a schedule for execution of each operator tree based upon dependencies among operators of each operator tree and resources available for executing the query including separating a resource intensive operator into a plurality of operators; and

returning a result indicating the operator tree having lowest execution cost based on its schedule for executing the query with available resources.

49. (Original) The method of claim 48, wherein the query comprises a Structured Query Language (SQL) expression.

50. (Original) The method of claim 48, wherein said creating step includes creating an operator tree including parallel operators for execution of portions of the query in parallel.

51. (Original) The method of claim 50, wherein said parallel operators comprise iterators for applying predefined behavior to data.

52. (Previously presented) The method of claim 50, wherein said step of creating

an operator tree includes creating operators for tasks to be performed in executing the query and said parallel operators provides for executing said tasks in parallel.

53. (Original) The method of claim 50, wherein a parallel operator executes in parallel across a plurality of storage units.

54. (Original) The method of claim 50, wherein a parallel operator executes in parallel across a plurality of CPUs.

55. (Original) The method of claim 50, wherein a parallel operator provides for pipelining of intermediate results from a first set of operators to a second set of operators.

56. (Previously presented) The method of claim 48, wherein said creating step includes creating an operator tree using a partitioning property so as to partition data among operators.

57. (Original) The method of claim 48, wherein said creating step includes generating a cost vector for each operator tree.

58. (Original) The method of claim 57, wherein said cost vector includes as components a selected one or more of work done by a processor in a given time, execution time of an operator, and resource usage of an operator for a certain time period.

59. (Previously presented) The method of claim 57, wherein said creating step further comprises:

pruning a first operator tree having a cost vector costing more in each vector dimension than a second operator tree.

60. (Original) The method of claim 48, wherein said creating step includes creating a plurality of operator trees based at least in part on partitioning and multi-dimensional costing.

61. (Previously presented) The method of claim 48, wherein said adjusting step includes adjusting an operator tree for maximum number of threads available at compile time.

62. (Previously presented) The method of claim 48, wherein said operator tree includes parallel operators for executing portions of the query in parallel and said adjusting step further comprises:

adjusting said parallel operators if necessary based on available memory resources.

63. (Canceled)

64. (Original) The method of claim 48, wherein said generating step includes identifying pipelines in each operator tree.

65. (Original) The method of claim 64, wherein said generating step includes constructing a pipeline dependency tree based on dependencies among operators of each operator tree.

66. (Original) The method of claim 65, wherein said creating step includes determining order of execution of pipelines based on the pipeline dependency tree and available resources.

67. (Original) The method of claim 66, further comprising:

if resource usage of a particular pipeline is greater than resources available for the particular pipeline, splitting the particular pipeline into a plurality of pipelines.

68. (Original) The method of claim 67, wherein said step of splitting the particular pipeline includes adding operators for materializing the particular pipeline into a plurality of pipelines at intervals such that resource usage is evenly distributed over the

plurality of pipelines.

69. (Original) A computer-readable medium having processor-executable instructions for performing the method of claim 48.

70. (Original) A downloadable set of processor-executable instructions for performing the method of claim 48.